Apellido y Nombre: ______________________________

Carrera: ________ DNI: ______________________________

[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática
Algoritmos y Estructuras de Datos

Algoritmos y Estructuras de Datos.
1er Parcial. [25 de setiembre de 2008]

ATENCIÓN: Para aprobar deben obtener un **puntaje mínimo** del 50 % en clases (Ej 1), y un 60 % sobre las preguntas de teoría (Ej 4).

[Ej. 1] [clases (30pt)]

a) **[lista (20pt)]** Escribir la implementación en `C++` del TAD lista (clase `list`) implementado por punteros **ó** cursores. Los métodos a implementar son `insert(p,x)`, `erase(p)`, `next()/iterator::operator++(int)`, `list()`, `begin()`, `end()`.

b) **[pila-cola (10pt)]** Escribir la implementación en `C++` de los métodos `push`, `pop`,`front` y `top` de los TAD pila y cola (clases `stack` y `queue`), según corresponda.

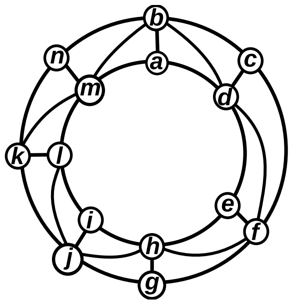
[Ej. 2] [Programación (total = 50pt)]

a) **[is-cyclic (30pt)]**
Dada una correspondencia M de enteros a enteros y un elemento x_0 podemos construir una secuencia $(x_0, x_1, ...)$ haciendo sucesivamente $x_1 = M(x_0)$, $x_2 = M(x_1)$,...,$x_{k+1} = M(x_k)$. La secuencia se detiene cuando el valor x_k no pertenece a las claves de M. Por ejemplo si $M = \{(1, 2), (2, 3), (3, 4)\}$ entonces dado $x_0 = 1$ se genera la secuencia $(1, 2, 3, 4)$. La secuencia se detiene en 4 ya que 4 no es una clave de M. En un caso así decimos que la secuencia generada es de longitud **finita**. Ahora bien, si alguno de los elementos de la secuencia, se repiten (digamos $x_{k+m} = x_k$, para algún $m > 0$) entonces a partir de allí es obvio que la secuencia se repetirá indefinidamente. Por ejemplo, si $M = \{(1, 2), (2, 3), (3, 4), (4, 2)\}$ y $x_0 = 1$ entonces la secuencia generada será $(1, 2, 3, 4, 2, 3, 4, 2, 3, 4, ...)$ es decir se genera una secuencia **cíclica infinita**.
Consigna: Escribir una función `void cyclic(map<int,int> &M,list<int> &L);` que extrae en `L` todas aquellas claves de `M` que generan una secuencia cíclica infinita. Por ejemplo, si $M = \{(1, 2), (2, 5), (3, 4), (4, 6), (5, 2)\}$ entonces `cyclic(M,L)` debe retornar `L=(1,2,5)`.
Sugerencia: Escrbir primero un *predicado* `int is_cyclic(map<int,int> &M,int x);` que determina si el elemento `x` genera una secuencia cíclica o no. Para ello se van guardando los elementos de la secuencia generada en una lista `Mx`. Cada nuevo elemento que se genera es comparado con los elementos de `Mx`, si el nuevo elemento generado está en `Mx` la secuencia es cíclica. Si la secuencia no es cíclica, entonces se detiene cuando algún elemento no pertence a las claves de `M`. *Nota:* Puede ser útil implementar y utilizar una función predicado `bool contains(list<int> &L,int x);` que devuelve `true` si el elemento `x` está en la lista `L`.

b) **[merge-map (20pt)]**
Dadas dos correspondencias A y B, que asocian enteros con listas ordenada de enteros, escribir una función `void merge_map(map<int,list<int>> &A, map<int,list<int>> &B, map<int,list<int>> &C)` que devuelve en `C` una correspondencia que asigna al elemento `x` la fusión ordenada de las dos listas `A[x]` y `B[x]`. Si `x` no es clave de `A`, entonces `C[x]` debe ser `B[x]` y viceversa. Por ejemplo:

$$A = \begin{cases} 1 \to (2, 3) \\ 3 \to (5, 7, 10) \end{cases} \quad B = \begin{cases} 3 \to (7, 8, 9, 11) \\ 5 \to (7, 10) \end{cases} \quad C = \begin{cases} 1 \to (2, 3) \\ 3 \to (5, 7, 7, 8, 9, 10, 11) \\ 5 \to (7, 10) \end{cases}$$

Sugerencia: Implementar y utilizar una función auxiliar `void merge(list<int> &L1,list<int> &L2,list<int> &L);` que devuelve en `L` la fusión ordenada de `L1` y `L2`.



[Ej. 3] **[color-grafo (5 ptos)]** Colorear el grafo de la figura usando el mínimo número de colores posible. Usar el algoritmo heurístico ávido. (*Nota:* debe recorrer los vértices en el orden que se indica, a, b, c, ...) ¿Puede indicar si la coloración obtenida es óptima? Justifique.

[Ej. 4] **[Preguntas (total = 15pt, 5pt por pregunta)]**

a) Ordenar las siguientes funciones por tiempo de ejecución:

$$\begin{aligned} T_1 &= \log_{100} n + \sqrt[3]{n} + 1.2 \cdot 10^n \\ T_2 &= 10\, 2^n + 4\, n! + n^2 \\ T_3 &= \sqrt{n} + 3 \log_2 n + 4\, n^2 + 2\, n^4 \\ T_4 &= 2^5 + 20 + 10 \log_2 n + 2 \log_{10} n \\ T_5 &= n^5 + 10\, 2^n + 10^{10} \end{aligned} \tag{1}$$

b) Comente una ventaja y una desventaja del uso de listas doblemente enlazadas con respecto a simplemente enlazadas.

c) Si la correspondencia `M={(1->2),(5->8)}` y ejecutamos el código `int x= M[5]`. ¿Que ocurre? ¿Que valores toman `x` y `M`? ¿Y si hacemos `x = M[3]`?

d) Cual es la complejidad algorítmica (mejor/promedio/peor) de las siguientes funciones:

1) `list<T>::begin()`,
2) `list<T>::insert(p,x)`,
3) `list<T>::clear(p,x)`,
4) `map<K,V>::find(x)`, para la implementación con vectores ordenados,
5) `map<K,V>::find(x)`, para la implementación con listas ordenadas.

e) Discuta si los algoritmos de búsqueda exhaustiva y heurístico para la coloración de grafos dan la solución óptima al problema. Señale la complejidad algorítmica de los mismos. ¿Cuál de los dos elegiría para un grafo con 100 vértices y 500 aristas? Justifique.